# ConTEXt 學習筆記

## Using MkIV

Athor: Li Yanrui

Email: lyanry@gmail.com

September 16, 2008

# 第 1 章　對 TEX 的一些認知

本章可讀，可不讀。讀，是爲了讓自己更明白 TEX 是什麼，並且瞭解一下 ConTEXt 的概況。不讀，是因爲取這樣標題的章節通常包含了許多廢話。

「TEX 是什麼」這一節，是 Hans Hangen 寫的[1]，我只是按照自己的理解翻譯了過來。之所以讓它在本章出現，是因爲我認爲作爲 ConTEXt 的主要開發者，Hans Hagen 對 TEX 具有深入客觀的認識，他有資格告訴我們一些 TEX 的眞實面目，而不是那些在網絡上流傳的諸多神話。另外，在本章中，我還記述了自己對 ConTEXt 的一些認識，並對這份文檔的寫作意圖進行些許交代。

## 1.1　TEX 是什麼

在此，我認眞思考了一些人提出的一些問題。這篇文章與其說它是一份評論，倒不如說它像一個開關，可以觸發更爲深刻地討論。如果你只是想知道什麼是 TEX，你可以直接讀「TEX 是什麼……以及我爲什麼喜歡它」那一節。

**所有的 TEX 都是平等的……**

**……只是有些 TEX 比其他 TEX 更加平等而已**

對於什麼是 TEX 這一問題實在很難給出明確的答案。例如 Peter Flom 使用「LATEX 是……」作爲開場白，將 TEX 程序等價爲 LATEX 這個宏包。這種等價隨處可見，因此許多用戶並不知曉 pdfTEX

[1] 原文見：http://www.tug.org/pracjourn/2005-3/walden-whatis/hagen-4.pdf

（程序）與 pdfLATEX（宏包）的區別。在許多系統上，如果沒有顯示的指明 pdfTEX 所使用的宏包，那麼它會調用 plain TEX 格式，這聽起來就更讓人混亂了。

TEX 的雙重身份也常常令人混亂，它即是一種排版語言，又是一個解釋器/排版工具。TEXBook 這本書中不僅講述了 TEX 的雙重身份，還涉及了 plain TEX 格式。因此，對於 TEX 而言，我們有必要區分：

**語言**

TEX 排版命令，由基本命令與宏命令構成，還有一些擴展命令，如 eTEX 添加的命令

**程序**

TEX（最早的TEX 解釋器）、pdfTEX、XƎTEX、LuaTEX 等 TEX 語言解釋器

**宏包**

Plain TEX、$\mathcal{A}\mathcal{M}\mathcal{S}$-TEX、LATEX、LA$\mathcal{M}\mathcal{S}$-TEX、ConTEXt 等基於語言實現的宏命令集

LATEX 用戶有多種調用 TEX 程序的方式：

**latex**

可預裝入 LATEX 宏包的 (pdf)TEX 引擎

**pdflatex**

同上，不過它是直接輸出 pdf 文檔

**xelatex**

由 XƎTEX 引擎預裝入的 LATEX 宏包

**lambda**

由 ALEPH 或 OMEGA 引擎預裝入的 LATEX 宏包

對於 ConTEXt 用戶而言，LATEX 調用 TEX 程序的經驗就用不上了，因爲 ConTEXt 提供了 texexec 脚本，可以使用統一的方式實現不同 TEX 程序的調用，譬如：

**texexec --pdf somefile.tex**

使用預裝入 ConTEXt 宏包的 pdfTEX 引擎

**texexec --xtx somefile.tex**

同上，不過這次調用的是 XƎTEX

很久以來，TEX 引擎產生 DVI 格式輸出，然後借助一些程序可以將其處理成 PostScript 文檔（適合打印或屏幕閱讀）。由於 PDF 文檔格式日趨流行，因此有必要再使用一些工具將 PostScript

文檔轉化爲 PDF 文檔。現在，使用 pdfTEX 引擎可以直接由 TEX 源文檔生成 PDF 文檔，所以也就沒有必要再延用過去的那種繁瑣的文檔生成方式了。

無論你怎樣使用 TEX,都應當搞清楚你所調用的命令的相關術語的確切概念，否則當你你與他人進行 TEX 知識交流或者去 BBS 尋求幫助時，別人未必使用與你相同的命令。比如在較早的 TEX 發行套件中，pdfTEX 默認輸出的文檔格式是 DVI，除非你顯示指定 PDF 格式輸出；而在新的 TEX 發行套件中，pdfTEX 默認輸出的文檔格式是 PDF。怎麼樣，現在還感到困惑麼？

**TEX 可以生成精美的文檔……**

**……但它不會給你什麼承諾**

有一些 TEX 的擁護者，他們爲了讓更多的人接受TEX，便經常地讚美 TEX 排版的精美。這種行爲過於一廂情願了。TeX排版的精美是沒有什麼疑問的，但是許多文檔看上去很 TEX 化，就像 MS Word 文檔看上去很 word 化，QUARK 文檔看上去很 quark 化。在風格的變化方面，字體及其格式的影響不是很大，而預定義的模板被重複地使用，這些都導致了文檔的版面千篇一律。例如，有些 TEX 用戶經常取笑使用 PowerPoint 演示文檔（因爲通過這些演示文檔的一些特徵就可以辨識出來這是 PPT 文檔），卻從未意識到他們自己也是那麼幹的。許多 TEX 用戶宣稱 TEX 可以有效地自動處理段落文本斷行問題，但是他們往往又不得不對那些很滑稽的行間距做出寬容的姿態，這些行間距通常是自己所使用的一些命令與那些嘗試一攬子解決所有問題的宏包發生衝突時所致。由於 TEX 在文本對齊方面做的非常出色，因此一旦文檔的版面發生單詞越出邊界的現象，就會非常得顯眼，在網絡上張貼的許多文檔經常向我們展示這一「特效」。

**TEX 是易於使用的……**

**……但需要你付出努力**

TEX 能夠很聰明地處理圖形與字體，但是郵件列表裡 (BBS) 的諸多帖子昭示著這並非是小事情，即便對於那些 TEX 老手而言。TEX 可以是一個易於使用的排版系統，但是用戶如果要眞正的駕馭它，也必須要經歷一個痛苦的過程。有些東西純粹是排版技巧，無關乎你使用的是哪一種排版系統。

使用 TEX 能得到一個極大的好處，那就是 TEX 用戶能夠普遍地熱心幫助新手。由於 TEX 用戶大都出於自己的意願選擇了 TEX，因而他們通常也能夠付出足夠多的努力來掌握它。

有關 TEX 及其宏包的用法，我們能夠在網絡或書店裡找到許多的學習資源（手册、答疑、Wiki、郵件列表、新聞組等等）。

**成也 TEX……**

**……敗也 TEX**

在許多計算機語言中，程序員不得不明確地告訴機器有一些文本要輸出。但是 TEX 卻不同，TEX 對於文本的任何修飾都可能會變成可見的。有人嘲笑 MS Word 排版的文檔會出現一些不協調的空區，譬如一些重複的空格。事實上，使用 TEX，如果你不瞭解一些宏標記的具體細節，也很容易引入許多很可笑的空區，導致頁面亂糟糟的。總之，在做一些宣判或開一些玩笑時，一定要小心謹慎才是。

在討論 MS Word 與 TEX 的區別時，有人說 MS Word 從來也搞不清楚在何種情況下需要切換字體，比如一個粗體顯示的單詞之前的空格是否也要粗體顯示？但實際上 TEX 在這方面並不比 MS Word 高明多少。

我們來看一下在 TEX 中如何能讓一個段落變的狹窄一些，如下：

```
\def\StartNarrow{\bgroup\leftskip1em\rightskip1em\relax}
\def\StopNarrow {\egroup}
\StartNarrow
some lines of text
\StopNarrow
```

像上面這樣的處理貌似可以將段落文本放置於一個已經作了限制的區域中。但實際上，上面的代碼並不能得到一個狹窄的段落，除非你顯式地添加上段落終止標記：

```
\StartNarrow
some lines of text\par
\StopNarrow
```

對於這一問題，最好的解決方案是修改 \StopNarrow 宏定義：

```
\def\StartNarrow{\bgroup\leftskip1em\rightskip1em\relax}
\def\StopNarrow {\par\egroup}
```

有許多間距都與採用這種方式得到的特徵以及並不總是很清楚的代碼效果有關。適用於這一份文檔的樣式並不見得就適合其它文檔。所有的一切都依賴於你的 TEX 是如何設定的以及宏包的作者如何協調好他們的工作。

**TEX 是穩定不變的……**

**……汗……我們真的希望是這樣？**

Don Knuth 一廂情願地認爲 TEX 程序的功能能夠適應性地擴展，去解決那些當前解決不了的問題。在優秀的老 TEX 中有兩個擴展的例子：特效 (Special) 與著述 (Write)。所謂特效，就是提供一種方法去控制 TEX 引擎以實現一些特殊效果，比如顏色或向圖形插入之類。如果沒有特效這一擴展，那麼 TEX 用戶就要面臨很大的麻煩，需要手動去做圖片複本的剪切、黏帖之類的事情。TEX 的著述擴展在撰寫文獻方面非常有用，它提供了目錄、交叉引用以及其他特徵，這些特徵的實現都需要一種反饋回路的運行機制。這兩種擴展都是以 TEX 宏包的方式實現的，因此 Don Knuth 滿懷信心地認爲 TeX引擎不需要修改，它可以通過宏包的形式不斷地充實自身的功能。

事實上，有一些非 Knuth 式的擴展，但並不是很多。因爲，鮮有人能不辭勞苦去寫一個用於化學領域文檔的子排版系統去與數學排版子系統並駕齊驅。許多擴展大都是採用 TEX 宏的形式來實現的。迄今爲止，沒有人對文本行號統計、並行輸出以及人性化及多語言兼容等方面提出健壯可靠的擴展方案。這又一次不得不借助 TEX 宏開發的方式來實現，這是一種很髒的方式。你可能大呼慶幸，因爲那些出版商沒有這些需求。

無論 TEX 引擎的實現有多麼完美，總是有人希望它能夠繼續改進。目前，最值得稱道的一些 TEX 擴展程序 $\varepsilon$-TEX、pdfTEX、XƎTEX。$\varepsilon$-TEX 提供了一些額外的編程功能。pdfTEX 將 TEX 推到了 21 世紀，提供了邊注字距調整與視覺縮放優化功能，另外還實現了完善的 PDF 輸出功能。XƎTEX 使

TEX 具備了處理 Unicode 編碼與 OpenType 字體的功能。實踐證明，只有不斷改進 TEX 引擎，才可以保證 TEX 不會被時代遺棄。[2]

當然，TEX 宏包也扮演了非常重要的角色。因爲無論 TEX 引擎怎麼變化，但這些宏包基本上還可以照常運行。這也就是說，很久以前寫的 TEX 文檔，利用擴展之後的 TEX 引擎還可以正常編譯輸出成適合印刷或適合屏幕閱讀的文檔。

有人讚美 TEX 系統鮮有 bug，但是對於 23 年之前使用 `\leaders` 排版命令的 TEX 文檔，使用現在的 TEX 引擎就無法再編譯了。因爲在這期間，TEX 引擎的一些 bug 得到了修正，因此對文檔的處理機制多少都有些變化，儘管這些變化非常之小。不過，通常而言，說 TEX 系統鮮有 bug 也不爲過，這種說法只是在你不刻意讓今天的 TEX 系統來處理很久很久之前的文檔或使用很久以前的宏包的前提下才成立的。順便說一下，也有一些有關程序穩定不變的例子，譬如計算機語言編譯器與解釋器，實際上 TEX 本身就是一種計算機語言 + 編譯器/解釋器。

**TEX 是什麼……**

**……以及我為什麼喜歡它？**

TEX 是一個允許你創建屬於你自己的排版環境的系統。在它所存在的這 20 多年來，出現了許多的排版環境，譬如 LATEX 與 ConTEXt，它們讓用戶可以更方便的使用 TEX。你可以根據自己的需要對它們進行功能上的擴展或者決定堅持使用它們所提供的功能，這完全取決於你個人。也有一些功能是你難以駕馭的，但這也是一個功能豐富的系統所帶給你的一個必然結果。

[2] 由於 Hans 撰寫這篇文章比較早，現在又有了一個新的 TEX 引擎---luaTEX，本文便是使用基於 luaTEX 引擎的 ConTEXt 編譯而得到的。luaTEX 項目與 XETEX 項目所要解決的基本問題是一樣的，但是前者實現了 Lua 語言在 TEX 引擎的嵌入，使得用戶更靈活地擴展 TEX 引擎的功能。

如果你能夠堅守著你所使用的排版環境的規範，比如保持你的文檔源碼清晰，那麼你的文檔就能夠耐得住時間的考驗。如果你採用結構化方式編輯文檔，以抽像的方式定義文檔的排版佈局，那麼最終可以實現在任何平台上都能夠得到最終的編譯輸出結果。你可以將文檔撰寫任務分配給其他人，你們一起協同工作，這種協同工作方式有助於實現郵件列表的支持，形成一個氣氛友好的社區、用戶群體，實現書籍與手册的撰寫。如果你想完全駕馭 TEX 系統（譬如 LATEX 或 ConTEXt），這需要耗費一段時間來掌握它們。學習週期過於漫長，這似乎是個難題，但是對於很對用戶而言，他們終生都在使用 TEX，並得益於此，所以學習週期的問題不再是問題。Don Knuth給予了我們創造精美文檔的能力，但是你需要付出一定的努力才能夠掌握它。Don Knuth 也給出了一個重要的時間界限條件，即在 100 年後，印刷技術有了極大的進步，這些使用 TEX 標記所寫的文檔依然可以使用 TEX 引擎進行有效處理。我們所寫的 TEX 文檔，能夠生存這麼長久，這本身就是一件很讓人舒服的事情。只是要小心，在這麼漫長的時間裡，你可能會碰到一些麻煩，要避免它們，就需要保持一種開放的心態去面對 TEX 的缺陷、神話以及一些奇怪的解決方案。

## 1.2　我對 ConTEXt 的認知

ConTEXt 是荷蘭 Pragma-ADE 公司基於 TEX 實現的一種高端文檔製造工具，使用它可以製作非常精美的 PDF 文檔，適用於科技文檔排版與屏幕演示文檔製作。與 LATEX 相比，ConTEXt 的開發更爲集中、活躍與激進。

ConTEXt 的版本可以分爲 MkI、MkII 和 MkIV。MkI 的用戶界面是荷蘭語，並且只有開發者可以看到用戶界面的具體實現。MkII 將用戶界面替換成英文，並且開放了一些用戶界面的實現，便於用戶參與開發。MkIV 是新一代 ConTEXt，其中許多模塊重新實現了，最具革命性的是引入了 luaTEX

引擎。luaTEX 是 pdfTEX 的一個擴展版本，其中植入了 Lua 語言，這意味著在 TEX 文檔中可以使用 Lua 完成一些程序，使 TEX 文檔演進成一種眞正的文檔排版編程語言。另外，luaTEX 提供對本地 TTF & OTF 字體的直接支持，對於中文用戶而言，困擾大家多年的中文字體嵌入的問題算是得到很好地解決。所以，luaTEX 似乎是可以結束目前 TEX 引擎版本混亂、功能落後的最理想的解決方案。

事實上 ConTEXt 還有一個 MkIII 版本，這是爲 XƎTEX 引擎預留的。XƎTEX 原本是 Mac OS 平台上的一個 TEX 引擎項目，不過現在 Linux、 Windows 平台都有其移植版本。XƎTEX 所要解決的問題與 luaTEX 差不多，但前者沒有像後者那樣提供一種內嵌的脚本語言。目前的 XƎTEX 已經可以較爲穩定地運行了，而 luaTEX 還處於 Beta 版本，據說今年夏天會正式發佈[3]。現在，XƎTEX 的最新版是 0.999，已經可以支持 UTF-8 編碼以及本地 TTF & OTF 字體調用，用於中文文檔處理基本上沒有什麼問題了。特別對於 LATEX 的中文用戶，由於 CTEX 論壇上的 mytex、yindian 所提供的 XeCJK 與 zhspacing 宏包，已經可以讓使用 XƎTEX 引擎的 LATEX 得以完美地支持中文排版。

現在，pdfTEX 項目已併入 luaTEX 項目中，這宣告著在今後一段很長的時間裡 Knuth TEX、luaTEX 與 XƎTEX 三足鼎立的時代的來臨，不過 Knuth TEX 引擎存在的意義也許僅在於兼容以前的文檔或留給後人去考古或者兼容歷史遺留文檔。

幾乎所有的 TEX 發行版中都包含了 ConTEXt 模塊，但是若想更容易地使用 MkIV 版本，建議安裝 ConTEXt Wiki[4] 上提供的 Minimals (ConTEXt 最小發行版)。ConTEXt Minimals 僅提供了運行 ConTEXt 環境所需要的軟件包，所以如果你想使用 LATEX，還請安裝其它 TEX發行版，譬如 TEX

[3] 在寫這份文檔的時候，今年夏天快要過去了。
[4] http://wiki.contextgarden.net

Live，只需不安裝其中的 ConTEXt 模塊即可，因爲 ConTEXt Minimals 與其它 TEX 發行版可以友好地共存。

## 1.3 關於這份文檔

這份文檔僅僅是我個人學習 ConTEXt 過程中所獲取的一些我認爲比較重要的知識的彙總，它也許有些凌亂，不是面面俱到，有些知識講述地過於淺薄甚至出現了認識錯誤，這都是因爲我還在學習中。如果您恰好是一個有經驗的 ConTEXt 用戶，恰好看見了錯誤，恰好又不吝賜教，你就是我的一字之師啊，（突然提高聲音）我記你一輩子！[5]

由於自知能力有限，我沒有將這份文檔寫成一部 ConTEXt 中文教程的慾望，因此這份文檔並不能教會你使用 ConTEXt。官方的文檔非常全面，如果想掌握 ConTEXt，我推薦你去閱讀它們。其實在這份文檔中，每當我碰到一些細節知識沒有耐心去講述時，便會偷懶，往往會裝作很耐心地告訴你在 ConTEXt 手册的哪一節可以找到詳細的記述。

我寫這份文檔的出發點與你之所以讀到它是一樣的，都是出於對 ConTEXt 的喜愛，並且希望它可以化作自己的如摶之筆，用以書寫優美的文檔。當我開始寫這份文檔的時候，還是一個菜鳥。待得這份文檔越來越完整，越來越豐滿的時候，也許我就變成了一個有經驗的菜鳥。我這麼說，不是謙虛，而是面對著 TEX 的博大，看見了自己的渺小。

本文檔有許多內容是只適於 Linux 環境，特別是關於 ConTEXt 編譯環境搭建方面的內容。Windows 或 Mac OS 環境中的 ConTEXt 我從未嘗試過，並且也不想去嘗試。在我看來，TEX 在 Linux 終端裡運行，再配合一些文本處理工具，儼然如魚得水一般，所以我一點都不嚮往 GUI 的陸地。

[5] 出自武林外傳第五十回。

# 第 2 章　讓 ConTEXt 工作起來

目前，只有 ConTEXt Minimals（ConTEXt 最小包）提供了最新的 ConTeXt 版本並且可以使之與系統所安裝的其它 TeX 發行版友好共存。本章介紹 ConTEXt Minimals 的安裝、基本使用以及 MkIV 的中文支持等知識，目的在於讓你多快好省地擁有一個 ConTEXt 文檔編譯環境。本章的最後介紹了一些初學者應當閱讀的文檔以及一些學習資源。

## 2.1　ConTEXt Minimals

首先，在本地機器上創建一個 ConTEXt 安裝目錄，該目錄的位置與名字視個人嗜好而定，我喜歡用 `/opt/context`。爲了不引起混亂，下文提及 ConTEXt 安裝目錄皆以如下設定的 Bash 變量 `$CTXDIR` 代替：

```
$ export CTXDIR=/opt/context
$ mkdir -p $CTXDIR
```

然後下載 ConTeXt 最小包安裝脚本，將其存放於 `$CTXDIR` 目錄並執行：

```
$ cd $CTXDIR
$ rsync -ptv \
rsync://contextgarden.net/minimals/setup/linux/first-setup.sh .
$ ./first-setup.sh
```

由於安裝脚本需要聯網下載 ConTEXt Minimals，需要等待一段時間才可以完成安裝。這段時間，你可以做點自己喜歡做的事情。

ConTEXt Minimals 安裝完畢後，就開始進行 ConTEXt 運行環境的配置，這需要設置許多的系統環境變量。不過莫要擔心，ConTEXt Minimals 提供了一個配置脚本，使用 source（亦稱點命令）命令加載該文件即可完成所有 ConTEXt 所需環境變量的設定。

```
$ . $CTXDIR/tex/setuptex $CTXDIR/tex
```

上述命令行可將 setuptex 配置文件中的諸多環境變量的設定在當前終端環境中保持有效性。這意味著，每次使用 ConTEXt Minimals，可以開一個終端窗口，然後執行一次上述命令，之後就可以在該終端中使用 ConTEXt 環境。如果需要使用系統所安裝的其它 TEX 發行套件，可另建一個未開啓 ConTEXt 環境的終端即可。

雖然每次啓用 ConTEXt Minimals 所輸入的命令行並不複雜，但是盡力讓命令行簡約且易於輸入一向是 Unix-like 系統的光榮傳統，因此我在 `$HOME/myscript` 目錄下建立了一個配置文件 ctx，該文件內容如下：

```
export TEXDIR=/opt/context/tex
source $TEXDIR/setuptex $TEXDIR
export OSFONTDIR="/usr/share/fonts/{adobe,winfonts}"
```

然後將 `$HOME/myscript` 目錄添加到系統環境變量 `$PATH` 並使之生效：

```
$ echo "PATH=$PATH:$HOME/myscript" >>  /.bashrc
$ .  /.bashrc
```

這樣，以後每次啓用 ConTEXt 環境時，只需執行以下命令：

```
$ . ctx
```

注意，在 ctx 文件中設置了 $OSFONTDIR 變量，該變量的意義在下一章講述 MkIV 中文排版時會進行詳細闡述，現在可不予理會。

使用 ctxtools 命令並配合 -updatecontext 參數可對所安裝的 ConTEXt Minimals 進行在線升級，不過前提是要建立 $TEXMFLOCAL 目錄。因爲 ConTEXt Minimals 升級包是一個 zip 壓縮檔，升級時會將該壓縮檔下載到 $TEXMFLOCAL 目錄並進行解壓縮。

ConTEXt Minimals 的升級很簡單，那就是再重裝一遍。

## 2.2 若需要 TEX 與 LATEX

ConTEXt Minimals 可以與其它 TEX 發行套件友好共存。最近，我打算使用 XƎTEX + plain TEX 格式寫一份文檔，但是 ConTEXt Minimals 默認未提供 XƎTEX 的 plain TEX 格式並且缺乏一些 plain TEX 必須的字體，因此基於 XƎTEX 的 plain TEX 及 LATEX 環境均無法正常運行。對於這個問題，我能想到的比較輕省的解決方案是安裝一個 TEX Live 2008，使用它所包含的 plain TEX 環境，另外再安裝一個 LATEX 環境以備不時之需。

首先從清華的 CTAN 鏡像 FTP 服務器下載 TEX Live 2008 的網絡安裝程序並解包：

```
$ wget ftp://ftp.tsinghua.edu.cn/mirror/CTAN/systems/texlive/\
tlnet/tldev/install-tl-unx.tar.gz
$ tar -zxvf install-tl-unx.tar.gz && cd install-tl-unx
```

TEX Live 2008 的安裝程序兼顧了不同用戶的軟件使用習慣，同時提供了文本界面與圖形界面兩種安裝模式：

```
$ install-tl   # 文本界面安裝
$ install-tl --gui # 圖形界面安裝模式
```

install-tl 程序在運行時會自動訪問預設定的 CTAN 鏡像 FTP 服務器，如果連接失敗，安裝過程也就終止了。對於國內用戶，可以通過 --location 參數將 CTAN 鏡像服務器設爲清華的 FTP：

```
$ install-tl --location=ftp://ftp.tsinghua.edu.cn/\
mirror/CTAN/systems/texlive/tlnet/tldev
```

如果網速足夠，那麼應當很快就可以在終端中顯示文本安裝界面了，具體的安裝過程應當參考 TEX Live 2008 中文指南[1]。TEX Live 2008 成功安裝後，只要不在終端中開啓 ConTEXt Minimals 的 ConTEXt 環境，就不會影響它的使用。

## 2.3 Hello World

將下面代碼使用文本編輯器保存爲 hello-world.tex 文件，這就是一個最簡單的 ConTEXt 源文檔。

```
% hello-world.tex
\starttext
Hello World.
\stoptext
```

[1] http://tug.org/texlive/doc/texlive-zh-cn/

我們可以在終端中使用 `context` 命令將這份源文檔編譯、輸出爲 PDF 文檔：

```
$ . ctx
$ context hello-world
```

上述命令將在當前目錄產生 hello-world.pdf 文檔，使用 PDF 閱讀器查看它，可以看到文檔中除了「Hello World.」字串之外，就剩下一個置於頂端的頁碼了。

在編譯 hello-world.tex 時，生成了許多中間文件，如果你不喜歡看到它們，可以使用以下命令將其清除：

```
$ context --purge
```

## 2.4 TEX 與 ConTEXt 的關係

在上一節 hello-world.tex 文檔的編譯過程中，我們可以獲得的直觀認識如下圖所示：


圖 2.1 ConTEXt 文 檔 編 譯 過 程 的 直 觀 認 識

實際上是沒有所謂的「ConTEXt 程序」的，眞正實現 TEX 文檔編譯的的是 TEX 引擎。在 ConTEXt MkIV 中，`context` 程序只是一個脚本文件，只有一行代碼：

```
mtxrun --script context "$@"
```

`mtxrun` 是一個 Lua 脚本，它需要調用 luaTEX 引擎並讀入 cont-en.fmt 文件才可以對 ConTEXt 文檔進行編譯處理。cont-en.fmt 是 ConTEXt 的格式文件，可以將它理解爲一本詞典，當編譯 ConTEXt 文檔時，LuaTEX 引擎遇到文檔中的排版命令時，就去 cont-en.fmt 文件中查找這些排版命令的定義並依命行事。可還記得 2.3 節中使用 `context --make` 命令麼？這個命令就是用於產生 cont-en.fmt 之類的格式文件的。這就是爲什麼我們經常說 ConTEXt 或 LATEX 是 TEX 的一種格式的原因。

## 2.5 讓 MkIV 講中文

要讓 ConTEXt MkIV 講中文，首先要讓它能夠找到中文字體所在。還記得 **2.1**節中我們寫的那個 ConTEXt 環境配置文件 ctx 麼？其中有一行代碼如下：

```
export OSFONTDIR="/usr/share/fonts/{adobe,winfonts}"
```

`$OSFONTDIR` 可以讓 LuaTEX 知道系統所安裝的字體所在。

我在 `/usr/share/fonts/adobe` 目錄中存放了 4 款 Adobe OTF 中文字體，在 `/usr/share/fonts/winfonts` 目錄中存放了一些從 Windows XP 中複製過來的一些中文字體。

有了中文字體，還需要一份字體配置文件。ConTEXt 提供了一套比較高級的字體機制，叫做 Typescript，它可以設置三種字型：襯線 (Serif)、非襯線 (Sans)與等寬 (Mono) 字型。這三種字型均

爲英文字體機制中的術語，就像我們中文字體有楷體、宋體、黑體等類型一樣。ConTEXt 的 Typescript 機制牽涉太多的字體知識，對它們的詳細描述已超出我之所能，這裡僅給出一份我照貓畫虎搞出來的的一份字體配置文件——zhfonts.tex，它是隨這份文檔一起發佈的，只需將它放到 TEX 目錄樹中，譬如：

```
$ mkdir -p $TEXMFLLOCAL/tex/context/third
$ mv zhfonts.tex $TEXMFLLOCAL/tex/context/third
```

然後執行：

```
$ context --generate
```

刷新一下文檔數據庫，就可以使用該字體配置文件了。

在 zhfonts.tex 中，我使用了 AdobeSongStd-Light、AdobeHeitiStd-Regular 以及 AdobeKaitiStd-Regular 字體作爲主要的中文字體。如果你沒有這些字體，可以將其替換爲其它 TTF 或 OpenType 中文字體，但是要保證 luaTEX 可以找到它們。由於中文字體所包含的英文字形通常比較醜陋，因此在 zhfonts.tex 文件中利用了 MkIV 的虛擬字體機制，將 lmroman10-regular、lmsans10-regular、lmmono10-regular、lmmono10-italic 以及它們的粗體的英文字形區域分別注入到相應的中文字體中。

現在，將 **2.3**節中的 hello-world.tex 文檔修改爲：

```
\usetypescriptfile[zhfonts]
\usetypescript[myfont]
\setupbodyfont[myfont,rm,11pt]
\starttext
```

```
世界，你好！
\stoptext
```

重新編譯這份文檔，即可得到中文版的 hello-world.pdf 文檔。

`\usetypescriptfile` 命令用於加載 typescript 文件，要求 typescript 文件與 helloworld.tex 文件均在同一目錄或者前者位於 ConTEXt Minimals 的某個可以檢索到的目錄中。

`\usetypescript` 命令表示使用 typescript 文件中定義的 typeface。所謂 typeface，就是一個字型族。在 zhfonts.tex 文件中所定義的 typeface爲 myfont，該字型族包含了 3 種字型：Serif、Sans 與 Mono。`\setupbodyfont` 命令是用於定義 hello-world.tex 文檔的默認字體，也就是正文字體，這裡是將 myfont 字型族中的 Serif 字型作爲文檔的默認字體，並且默認尺寸爲 11pt。

現在，MkIV 基本上已經解決了中文排版問題，我們應當爲此感謝 Hans、Taco 等開發者的辛勤勞動。另外，還要感謝 Wang Yue、SDE 諸位同學，他們提出了許多有關 ConTEXt 中文支持的要求並報告了許多 bug。特別是 Wang Yue 同學，自去年始，一直在關注 LuaTEX & ConTEXt MkIV 等項目的發展，並積極地與開發者們溝通，使得開發者認眞考慮了有關中文處理的諸多需求。

## 2.6 開始 ConTEXt 的旅程

ConTEXt MkIV 中文支持的成功嘗試，奠定了我開始學習 ConTEXt 的信心。ConTEXt 的文檔非常豐富，不過大部分都是英文的，這多少讓我有點沮喪。雖然讀英文文檔不是什麼大障礙，但是總沒有中文文檔來得親切。現狀如此，只好忍了。

目前，ConTEXt 的大多數文檔是針對 MkII 版本的，除了 MkIV 手冊之外，似乎再也沒有其它文檔來講述 MkIV 的應用。不過，由於 MkIV 改變的主要是 ConTEXt 底層，用戶界面基本上沒有改

變，對於中文用戶而言，基本上只需將 MkII 的中文字體支持機制改換爲 MkIV 的即可，這樣原來適用於 MkII 的文檔基本上也適用於 MkIV。現在，ConTEXt 項目組現在正在組織進行新的 ConTEXt 文檔撰寫，這個工程非常浩大，其意在於將這麼多年零散的文檔彙總到一起，並追隨 ConTEXt 的最新進展，估計要到明年方能竣工。遠水解不了近渴，現在學習 ConTEXt，還是要從 MKII 版本的文檔開始。

作爲初學者，*ConTEXt, an excursion*[1]文檔是官方製作的新手入門必讀文檔。ConTEXt 的官方文檔通常是分爲屏幕閱讀版本與打印版本，我讀的 *ConTEXt, an excursion* 是屏幕閱讀版本，第一次打開這份文檔，就驚愕了一下，沒想到 PDF 文檔居然可以做得如此精緻。內容淺顯易懂，文檔版面美觀，我覺得 ConTEXt 是非常禮遇初學者的。

當讀過 *ConTEXt, an excursion* 時[2]，就可以使用 ConTEXt 排版手頭上正在寫作的一些文檔，其實這才是眞正的開始學習 ConTEXt。對於其它任何一種 TEX 都是如此，只有眞正開始使用它，才有可能掌握它。一開始，文檔的版面醜點沒關係，畢竟內容是最重要的。隨著對 ConTEXt 的認識日益深刻，你的文檔便會愈發具備可觀賞性。在實踐的過程中，ConTEXt 用戶手册[2]是主要的參考書。當你遇到不解的命令/參數，或者想瞭解一些命令更多的細節，都可以使用 PDF 閱讀器打開 ConTEXt 用戶手册，利用閱讀器提供的查詢功能找到你所感興趣的內容，並掌握它們。

字體在排版中是決定版面美觀的重要因素，閱讀 ConTEXt 新手册中已經基本完工的 *fonts*[3] 部分，便可以理解 ConTEXt 的字體機制，你將獲得可以在 ConTEXt 自如使用你喜歡的字體的能力；同時，我製作的那份 MkIV 中文字體配置文件也不再會令你感到困惑。

[2] 實際上我是用到什麼就去看什麼，現在大概看了有一半內容。

若使用 ConTEXt 排版時存在向量圖繪製需求，推薦使用 METAPOST。不過，在 ConTEXt 中，我們要面對的不是 METAPOST，而是 MetaFun[4]，後者對前者進行了封裝，實現了 ConTEXt 與 METAPOST 的完美結合。*ConTEXt, an excursion* 文檔版面之所以美觀，是因爲在排版中大量使用 MetaFun來美化版面佈局的結果。

若你對 ConTEXt 已經有所認識，那麼 ConTEXt Wiki[3] 是非常好的網絡學習資源。若有能力，可以在該 Wiki 上與整個世界的 ConTEXt 用戶分享你的學習經驗；若暫時沒能力，可以選擇潛水。目前，我是這個 Wiki 衆多潛水員中的一名。

目前，ConTEXt MkIV 也許還存在許多的 bug，當你發現它們，應當提交 ConTEXt 郵件列表[4]中；當你使用 ConTEXt 時有不解的問題，也可以去郵件列表中尋找答案。說來慚愧，我英文不好，並且不熟悉郵件列表的交流規則，所以一直都膽怯於郵件列表上的交流。

介紹了以上學習資源之後，我開始如釋重負。以後再在這份文檔上摺騰，心裡就不會總是想著自己在寫一本 ConTEXt 教程，那樣子很累，並且很容易喪失樂趣。這份筆記的眞正的意圖是自我備忘，將它投放到網絡上，是希望我的經驗能夠對他人有所幫助並且能夠一同學習、討論 ConTEXt 排版知識。

Just for fun!

[3] http://wiki.contextgarden.net/Main_Page
[4] http://archive.contextgarden.net/list/context.en.html

# 第 3 章　版面設計

其實我不懂排版，我只是希望這份文檔的版面看上去不那麼醜陋，所以我經常留意有關排版的知識，並樂於使用 ConTEXt 來實踐。我將這個實踐的過程整理出來，就是本章所記述的內容，並且每當我修改本文檔的版面風格時，這一章的內容也會適應性的變動。可以將本章看作是這份筆記版面設計的說明書。

## 3.1　頁面與紙張

頁面尺寸就是文檔的版面尺寸，而紙張尺寸是指文檔印刷用紙的規格，它們的取值一般是相同的；只有在要求每張印刷用紙上實現多頁排版時，二者的取值才不一樣。這份筆記的屏幕閱讀文檔頁面與紙張尺寸設定如下：

```
\definepapersize[SCREEN][width=24cm,height=18cm]
\setuppapersize[SCREEN][SCREEN]
```

其意是指頁面與紙張的尺寸的寬高分別爲 24cm 與 18cm，這是考慮到顯示屏幕尺寸的寬高比一般爲 4:3，這樣可以充分利用屏幕區域，並且翻頁方便。

頁面與紙張的規格可以採用 ConTEXt 已經預定義好的國際標準規格，如 A0 ~ A10、B0 ~ B5 等；也可以採用自定義尺寸：

```
\definepapersize[I18][width=18.5cm,height=23cm]
\setuppapersize[I18][I18]
```

這裡定義的 I18 規格的頁面與紙張尺寸，是近年來計算機技術書籍排版常用的紙張規格。

## 3.2 版面佈局

ConTEXt 將版面劃分爲 25 個區域，具體的區域劃分在 *ConTEXt, an excursion*[1]文檔的第 32 章 *Page Layout* 中可以看到，也可以在 ConTEXt Wiki 的 *Layout* 頁面看到[1]。

我爲這份文檔所規劃的版面佈局如下：

```
\setuplayout
  [width=fit,
  height=middle,
  leftmargin=3cm,
  rightmargin=3cm,
  backspace=4cm,
  topspace=.5cm,
  headerdistance=.4cm,
  footerdistance=.4cm,
  header=1cm,
  footer=1cm]
```

下一頁的插圖顯示了這些參數的佈局效果，這裡交代一下，那幅插圖是利用了 Patrick 寫的一個模塊 (Module)[2]繪製的。

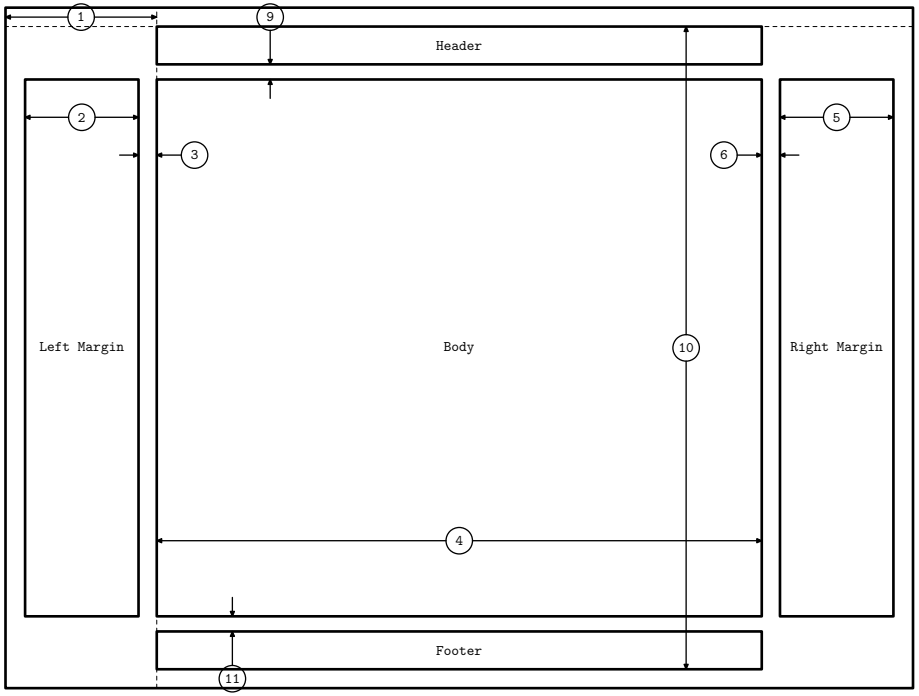
[1] http://wiki.contextgarden.net/Layout
[2] http://modules.contextgarden.net/t-layout

`\setuplayout` 命令所具有的參數，在數量上堪稱是 ConTEXt 命令之最，如果我沒有數錯，應該是 44 個，常用的參數也就是下一頁插圖中所示的那些了。通過這些參數，ConTEXt 可實現各種複雜的版面佈局。

說點閒話。如果我沒記錯的話，MS Word 用來設置版面佈局的參數不超過 10 個。在 LATEX 中，需要借助一些宏包才可以比較方便地實現版面佈局控制。複雜與統一，也許這是 ConTEXt 的一個很重要的特徵，複雜是追求功能強大的必然結果，而統一可以讓用戶只需一份官方提供的技術手冊就可以毫無爭議地使用這些功能。

注意，將 `\setuplayout` 的 `width` 與 `height` 參數設爲 `fit` 或 `middle` 都可以讓 ConTEXt 自動爲我們確定它們的值。`fit` 參數的確定是根據周邊鄰域參數計算出來的，而 `middle` 不是很關心周邊參數，它只需要將 `width` 或 `height` 所確定的尺度相對於同一維向的頁面尺度是居中的。

`location` 這個參數讓我費解了許久，歷經多次嘗試，總算知道一點所以然。當我們將版面尺寸與紙張尺寸設爲相同時，`location` 這個參數無論如何設定都是看不到效果的。當紙張尺寸大於版面尺寸時，`location` 的值可以表示版面在紙張的位置，譬如 left、right、middle、singlesided 以及 doublesided。




1 backspace 113.8 pt
2 leftmargin 85.4 pt
3 leftmargindistance 13.7 pt
4 width 455.2 pt
5 rightmargin 85.4 pt
6 rightmargindistance 13.7 pt
7 topspace 14.2 pt
8 header 28.5 pt
9 headerdistance 11.4 pt
10 height 483.7 pt
11 footerdistance 11.4 pt
12 footer 28.5 pt
paperwidth 682.8 pt
paperheight 512.1 pt

## 3.3 頁眉、頁腳

對於這份文檔的頁眉，我將奇數頁的頁眉設爲當前章標題（左）+ 頁碼（右）；偶數頁的頁眉設爲當前節標題（左）+ 頁碼（右）：

```
\def\CurrentChapter{%
    第 \headnumber[chapter]\ 章%
    \hbox to wem{}%
    \getmarking[chapter]%
}

\def\CurrentSection{%
    \headnumber[section]%
    \hbox to 2em{}%
    \getmarking[section]%
}

 \setupheadertexts
    [\CurrentChapter][pagenumber]
    [pagenumber][\CurrentSection]
```

\setupheadertexts 命令的參數設置的有些古怪，按手册上講的，前兩個參數分別用於設置奇數頁的頁眉左側與右側內容，後兩個參數則用於設置偶數頁的頁眉左側與右側內容，但是在上述的頁眉設置語句中，我只有將偶數頁的頁眉內容的左右位置掉換過來才可以得到這份文檔的頁眉結構。我並不知道爲什麼要這樣設置，只是多次嘗試才知道只有如此才能實現我的要求。

由於設置在頁眉中顯示頁碼，所以應當引去 ConTEXt 自動爲頁面添加的頁碼，順便把頁碼的字號也設置一下：

```
\setuppagenumbering
    [style=\tfx,location=]
```

我感覺 ConTEXt 是有點另類，居然讓 \setuppagenumbering 來設置雙面打印文檔模式。如果，我想將本文檔的頁眉結構變成：奇數頁的頁眉設爲當前章標題（左）+ 頁碼（右）；偶數頁的頁眉設爲當前節標題（右）+ 頁碼（左），只需在 \setuppagenumbering 命令中設定 alternative 參數即可：

```
\setuppagenumbering
    [alternative=doublesided,style=\tfx,location=]
```

感覺有點莫名其妙，也許因爲我還未發現正確的頁眉設置方法的緣故。

頁脚的設定方法與頁眉類似，我在頁脚的左側設置了顯示這份文檔的名稱，在其右側顯示可以跳往目錄頁的鏈接：

```
\startmode[screen]
\setupfootertexts
    [{\ConTeXt\ MkIV 學習筆記}][{\goto{回目錄}[content]}]
    [{\goto{回目錄}[content]}][{\ConTeXt\ MkIV 學習筆記}]
\stopmode
```

用到了 ConTEXt 的 mode 環境，這就像 C 語言中的編譯宏。只有在 ConTEXt 編譯命令中設定對應的 mode 值，對應 mode 環境中的語句才是有效的。要讓 screen mode 環境生效，可：

```
$ context --mode=screen ctxnotes
```

ConTEXt 的 mode 環境非常有用，許多 ConTEXt 官方製作的文檔都是利用 mode 環境同時產生屏幕閱讀版本與打印版本。將來，我打算將這一環境用於製作演示文檔與講稿。

在頁脚右側顯示的「回目錄」的鏈接需要設置目錄代碼的支持，因爲這需要利用 ConTEXt 的引用機制：

```
% 我的目錄頁
\title[content]{目錄}
\placecontent
```

有關文檔目錄設置的細節，準備單獨開一個章節來記述。

## 3.4 封面設計

ConTEXt 沒有像 LATEX 那樣提供了專用於封面製作的命令，只是提供了一個 standardmakeup 環境，該環境的作用就是建立一個不增加頁碼計數的頁面，使得用戶可以在其中排版封面、扉頁之類的頁面。這份文檔的封面的具體設計如下：

```
\startstandardmakeup
  \startcolor[mydarkred]
  \switchtobodyfont[14pt]
  \hfil\bfd\ConTeXt 學習筆記\hfil
  \blank[1.5cm]
  \hfil\bfc Using MkIV\hfil
  \blank[12cm]
```

```
    \startalignment[flushright]
      \ssa
      \bTABLE
        \setupTABLE[r][each][frame=off]
        \bTR \bTD Athor: Li Yanrui\eTD \eTR
        \bTR \bTD Email: lyanry@gmail.com\eTD \eTR
        \bTR \bTD \date \eTD \eTR
      \eTABLE
    \stopalignment
    \stopcolor
  \stopstandardmakeup
```

請寬恕我在上述語句中很可恥地使用了 1.5cm、12cm 這樣確定的長度。因爲我還不怎麼會駕馭 ConTEXt，還無法靈活地實現可根據版面的變化自適應調整的封面設計。

要設計一個美觀的封面眞得很難，而且這也不是我所擅長的。現在，擁有這樣一個簡單的封面，我已經心滿意足。也許等我以後漸漸熟悉 MetaFun 之時，會讓它再美觀一些。

## 3.5 標題樣式

慣見的文檔章、節標題一般都是採用黑體，字號通常也比正文字號大一些，另外標題前後也留出了一些垂直間距。這些設置無非是爲了告訴讀者，它們是標題，而不是正文。這份文檔的標題樣式設置如下：

```
\setupheads[indentnext=yes]
\setuphead
    [chapter]
    [style=\bfc,header=empty,footer=empty]
\setuphead
    [section]
    [style=\bfa]
\setuphead
    [title]
    [style=\bfb,header=empty,foote=empty]
\setuphead
    [subsubject]
    [style=\bf]
```

\setupheads 命令可以設置所有類型的章節標題默認狀態。在設置章標題時，由於其所在頁面通常不需要頁眉與頁脚，所以 header 與 footer 都應設爲 empty。

現在使用 MkIV 排版中文，要想實現章節的編號格式爲「第 x 章」、「第 x 節」，或者實現圖、表標題格式爲「圖 x」、「表 x」，可能需要自己來做一些有些 dirty 工作。ConTEXt 提供了 \setuplabeltext 命令，可用於設置表格、圖、章節、附錄的編號格式。譬如，要設置中文的章節編號：

```
\setuplabeltext [en] [chapter={第\;,\;章}]
\setuplabeltext [en] [section={第\;,\;節}]
```

遺憾的是，我不知道怎樣將編號中的數字也使用中文來表示，這在 MkII 中是可以的。

## 3.6 段落與行

中文排版，段落間距通常等於行間距，因爲中文段落是通過首行縮進來標示的，下面代碼可將段落文本首行縮進 2 個漢字距離：

```
\setupindenting[always,2em,first]
```

ConTEXt 默認是將段間距設置爲 0，這碰巧符合中文的排版規範。如果有設定段間距的需求，就查一下 `\setupwhitespace` 命令的用法。

對文檔正文而言，行間距的設置對於排版的美觀性是至關重要的：行間距過小則閱讀困難；行間距過大又顯得版面稀疏，浪費紙張。ConTEXt 默認的文本行間距的值對於中文排版而言過小，可通過 `\setupinterlinespace` 命令結合自己的審美嗜好對行間距進行調整。我是這樣設置行間距的：

```
\setupinterlinespace[big]
```

除了 `big` 參數之外，`\setupinterlinesapce` 命令還可以接受 `small`、`medium` 參數，但我以爲只有 `big` 適合作爲中文文本行距。值得注意的是，`small`、`medium` 和 `big` 並不表示眞正的行間距尺寸，它們分別表示的正文字體尺寸的 1.0、1.25 和 1.5 倍。這份筆記所使用的正文字體是 11pt，那麼行間距就是 16.5pt。

# 第 4 章　引用



當我用我這個代號來進行對話的同時，你的代號也是我，這意味著什麼呢？這是否意味著你就是我，而我也就是你？

## 4.1　目錄

如果不那麼在乎目錄的樣式，可以在期望插入目錄之處添加：

```
\completecontent
% 或者
\placecontent
```

`\completecontent` 與 `\placecontent` 的區別就是前者是生成的目錄列表頁面是帶有標題的；而後者只生成目錄列表，如果需要頁面帶有標題，可以使用 `\title` 之類的無編號的標題命令實現:

```
\title{目錄}
\placecontent
  [alternative=a,
   style=normal,
   numberstyle=bold]
```

在章標題之後使用 \placecontent：

```
\chapter{引用}
\placecontent
  [alternative=a,
   style=normal,
   pagenumber=no,
   margin=2em]
```

可以實現在當前章標題下面顯示出這一章所有小節的列表。只是比較怪異，條目的顏色居然不一樣。好在使用自由軟件多年，我已經能夠容忍這些小問題了。

由於我比較喜歡 ConTEXt 手册的那種目錄樣式，便從手册的源碼中找到了該樣式的具體設置，然後照虎畫貓一番，並添加了一些小創意：

```
\def\ChapterNumber#1{\doiftext{#1}{第\;#1\;章\quad}}
\setuplist
  [chapter]
  [alternative=a,
   before={\page[preference]\blank},
   after=\blank,
   style=bold,
   width=fit,
   pagestyle=boldslanted,
   pagenumber=no,
   numbercommand=\ChapterNumber]
```

```
\def\PageNumber#1{\color[darkgray]{#1}.}
\setuplist
  [section]
  [alternative=d,
   style=small,
   pagecommand=\PageNumber,
   pagestyle=\itx]
```

結果便是這份文檔目錄頁中所顯示的那個樣子。我所說得小創意，實際上就是讓章節編號變成半中文化（因爲還夾著阿拉伯語呢）；另外就是讓目錄列表中各小節的頁碼顯示爲灰色，不那麼搶眼。

## 4.2 交叉引用

交叉引用實在是太有用了，即便也不怎麼經常用到它，也還可以借此嘲笑一下 Word 交叉引用功能的孱弱，贏得一次口水戰的勝利。

這份文檔的頁脚的右側的「回目錄」鏈接就是通過交叉引用來實現的，在**3.3**節（瞧，這就是一個交叉引用）中的頁脚設置：

```
\setupfootertexts
    [{\ConTeXt\ MkIV 學習筆記}][{\goto{回目錄}[content]}]
    [{\goto{回目錄}[content]}][{\ConTeXt\ MkIV 學習筆記}]
```

其中，\goto 這個命令可以跳到任何設置了引用的文檔位置。這裡，它要跳到 content 引用所在的位置，由於這個引用我是在設置目錄頁面時設定的：

```
\startfrontmatter
  \setuppagenumbering
    [conversion=romannumerals]
  \title[Content]{目錄} % 在此設定了引用
  \placecontent
\stopfrontmatter
```

因此 \goto 會帶你跳到目錄頁。目錄頁面的引用是通過 content 這個名稱標識的，\goto 命令可以根據這個標識確定所要跳往的頁面位置。

ConTEXt 許多排版命令都支持引用設置，譬如章節標題、插圖、表格等命令。一定要記住，文檔中只要出現諸如「第 xxx 章」、「xxx 節」、「圖 xxx」、「表 xxx」之類的內容時，這就意味著您應當使用交叉引用機制了。比如，我要引用當前這一節，首先要爲節標題命令設定引用名稱：

```
\section[cross references]{交叉引用}
```

然後在需要引用本節之處，通過 \in 命令即可獲取節號；如果是屏幕閱讀文檔，還可以讓節號作爲一個鏈接，指向本節所在頁面位置。同類的命令還有 \at 與 \about，前者可以獲取引用所在的頁面，後者可以得到引用的內容。下面仿照 ConTEXt 手册中的例子，演示這三個命令的用法：

```
我引用了第 \at[cross references] 頁的 \in[cross references] 節 \about[corss
references]的一些內容。
```

效果：我引用了第 **34**頁的 **4.2**節 **“交叉引用”**的一些內容。

現在，MkIV 還未能實現引用在頁面的準確定位，所以在屏幕閱讀文檔中的一些鏈接只是將你帶往這些引用所在的頁面而不是將準確的位置展示在你眼前。

## 4.3 索引

在閱讀 *ConTEXt* 手冊或者 *ConTEXt, an excursion* 之類的文檔時，在附錄中可以看到許多命令的索引，這些索引會告訴你那些命令在哪些頁面出現了；如果是屏幕閱讀文檔，還可以將頁碼變成指向對應頁面的鏈接。利用這些索引功能，我可以快速找到這些命令的細節知識。這在寫技術文檔時非常重要。在這份文檔中，我沒有使用索引的功能，因爲很少有人會爲了一份學習筆記如此大動干戈。將它記述於此，僅僅是爲了將來有可能使用到它。

要定義索引很簡單，使用\index 這個命令就可以：

```
我在這裡演示 index\index[index]{\tex{index}} 與
 placeindex\index[placeindex]{\tex{placeindex}} 命令的用法。

\placeindex
```

然後使用 \placeindex 或 \completeindex 命令在需要放置索引的頁面顯示索引。

索引列表默認是以字母進行分組與排序的，這對於西文文檔排版很適用，但是沒法處理中文。可以採用漢語拼音來解決這個問題。譬如：

```
……可以採用漢語拼音\index[HYPY]{漢語拼音}來解決這個問題……
```

生成的索引列表如下：

索引與目錄的用法很相似。實際上這一章中所有的內容都是基於 ConTEXt 的引用機制實現的。索引與目錄存在著更爲高級的用法，但是我沒有那麼多興趣和精力去折騰它們，以後若有這方面的需要再下手也不遲。

## 4.4 參考文獻

寫科技文檔必然離不開參考文獻 (Bibliography) 的支撐，所以要評價一個文檔排版軟件的優劣，其參考文獻管理功能是一項非常重要的指標。與 LATEX 類似，ConTEXt 主要是基於 BibTEX 維護參考文獻，這是借助 Taco Hoekwater 寫的 t-bib.tex 模塊來實現的。

### BibTEX

BibTEX 的主要功能是管理參考文獻信息並提供排版格式。在使用 BibTEX 之前，用戶需要建立參考文獻數據庫，不要害怕，這裡的數據庫實際上就是一個擴展名爲 .bib 的文本文件，我們在其中記錄文獻信息。下面是一份 .bib 文件示例：

```
@book{諸葛專著2008,
  author    = "諸葛亮",
  year      = "2008",
  title     = "木牛流馬製造工藝 [M]",
  publisher = "蜀國機械工業出版社"
};

@article{諸葛論文2008,
```

```
    author    = "諸葛亮",
    title     = "論伐魏的必要性與可行性 [J]",
    journal   = "蜀國軍事",
    volume    = "13",
    number    = "110",
    year      = "2007"
}
```

在文獻數據庫中，可以存儲許多種文獻類別，常用條目類型有 article、book、conference、manual、misc、techreport 等。每種參考文獻類別由多個域組成，有些是必須寫得，沒寫會給出警告，而有些是可選。譬如 book 類別中，不可省略的域有 author, title, journal, year，可省略的域有 volume, number, pages, month, note。推薦做法是在參考文獻數據庫中儘可能地提供文獻的詳細信息。關於 .bib 文件的格式說明，請參考 BibTEX 文檔[5]。

BibTEX 爲便於用戶管理參考文獻列表的排版風格，提供了 .bst 文件。.bib 文件與 .bst 文件之間的關係宛若 HTML 與 CSS 之間的關係，它們都遵守內在數據與外在表現分離的遊戲規則。用戶在 .bst 文件中可使用 BibTEX 定義的一種微型語言來實現對參考文獻列表風格的控制，但是在 ConTEXt 的 bib 模塊中，僅使用 .bst 文件控制參考文獻條目的排列次序。一般情況下，用戶不需要製作 .bst，所以這個瞭解一下就可以了。

BibTEX 與 ConTEXt 的協同工作時，需要 ConTEXt 文檔編譯命令輸出 .aux 文件。基於 .aux 文件，BibTEX 在 .bib 檢索所需的文獻信息，並結合 .bst 文件實現參考文獻列表外觀控制，最終輸出 .bbl 文件。一旦我們得到了 .bbl 文件（實際是 TEX 文檔），那麼就可以繼續進行 ConTEXt 文檔編譯過程，最終輸出帶有參考文獻信息的文檔，這一工作流程如下圖所示。

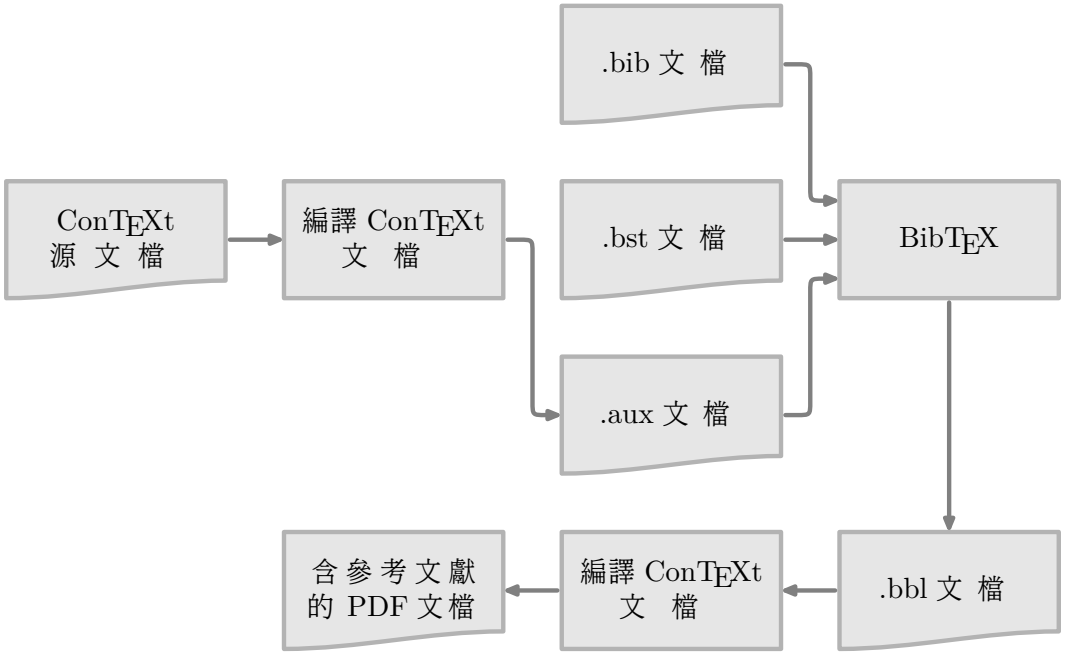



圖 4.1　BibTEX 工　作　過　程

### ConTEXt 的 bib 模塊

ConTEXt 是通過 Taco 寫的 bib 模塊取得 BibTEX 的協同，用戶可以在 ConTEXt 文檔中實現參考文獻列表排版樣式的設置。下面是一份最爲簡單的 ConTEXt 參考文獻示例文檔 `example/ex-4.tex`：

```
% 設置中文字體
\usetypescriptfile[zhfonts]
\usetypescript[myfont]
\setupbodyfont[myfont,rm,11pt]

\usemodule[bib] % 啓用 bib 模塊
\setupbibtex[database=example]

\starttext
這一年，諸葛亮同學出版了一本專著\cite[諸葛專著2008]，並在核心學術期刊上發表了一篇論文\cite[諸葛論文2008]，因此深得劉備的賞識。
\completepublications
\stoptext
```

上例中 \setupbibtex 命令的 database 參數的值爲 .bib 文件名，我是將上一節中的那個 .bib 文件命名爲 example.bib，然後在此使用。\completepublications 命令會在所排版的文檔中插入參考文獻列表。

編譯上述示例文檔的過程正如圖 4.1所示的那樣，需要三個處理步驟，如下：

```
$ context --once ex-4   # 產生 .aux 文檔
$ bibtex ex-4         # 產生 .bbl 文檔
$ context ex-4         # 完成文檔編譯
```

如果嫌每次編譯文檔都要重複輸入三次命令過於繁瑣，花上 20 分鐘學習一下 Makefile 的簡單用法會讓我們更輕鬆一些。

查看一下帶有參考文獻的 PDF 文檔，可能你會發現參考文獻並非是我們常見的那種風格，不要著慌，ConTEXt 提供了一些用於設置參考文獻風格的命令供我們使用。下面對 ex-4.tex 文件略作修改，添加了 \setuppublications 代碼：

```
...
\usemodule[bib] % 啓用 bib 模塊
\setupbibtex[database=example]
\setuppublications[alternative=num]
...
```

重新編譯一次 ex-4.tex 文檔[1]，現在參考文獻的樣式基本符合我們常見的風格了。有關 \setuppublications 命令的詳細用法請參考 bib 模塊文檔[6]。

### Zotero

Zotero 是 Firefox 的一款擴展 (Extension)，號稱是下一代研究工具[2]，我使用它來管理參考文獻數據。除 Zotero 之外，還有許多其它文獻管理軟件，比如 Endnote、RefWorks、jabref 等，與它們相比，Zotero 將我們進行文獻管理的整個過程——蒐集、整理、閱讀和引用等步驟都很好地集成在一起，並且與 Firefox 取得完美的結合，莫要忘記我們通常要在網上尋找文獻的。關於 Zotero 的詳情見其項目主頁[3]。

[1] 若是未有改動 .bib 文檔或者未引用新的文獻，只需使用常規 ConTEXt 文檔編譯方式即可。
[2] 至於上一代是誰，不可考，也許是那些脫離網絡環境的文獻管理軟件吧。
[3] http://www.zotero.org/

大多數情況下，使用 Zotero 的流程可以歸納爲：在互聯網上發現文獻——利用導入功能得到文件信息和有關附件——製作文獻閱讀筆記、tag 以及相關文件鏈接——搜索文獻——輸出有關文獻爲其他軟件格式——在 TeX 或字處理軟件中引用。除最後一步外都不需要打開任何其他軟件。

關於 Zotero 的具體用法……咳咳……實在不知道該如何用文字來講述一款 GUI 工具的使用，又不勝一副一副配圖介紹的那般繁瑣。自我開脫一下，畢竟這是一份個人學習筆記，而不是一份教材，所以推薦去看 Zotero 的操作視頻教程[4]。

### 設置參考文獻列表的樣式

在製作這份學習筆記的參考文獻時，因爲有一些文獻來自於網絡，我希望在參考文獻列表中提供它們的網址 (URL) 並且要求是超級鏈接格式的，這樣便於讀者直接驅動網頁瀏覽器去下載它們。按照 BibTEX 的 .bib 文檔語法，我採用以下文獻條目格式提供此類文獻的信息：

```
@booklet{Taco,
 title = {{Bibliographies}},
 author = {Taco Hoekwater},
 url = {http://modules.contextgarden.net/bib}
}
```

但是在文檔中去引用上述文獻時，其 URL 信息無法在參考文獻列表中顯示，更談不上超級鏈接格式了。在 ConTEXt Wiki 中的 Bibliography 文檔[7]中講述了如何在參考文獻中啓用 URL 的方法，

[4] http://www.zotero.org/videos/tour/zotero_tour.htm

並提供了一個示例。我對這個示例進行了一些簡化，貴在於理解其要義。將以下代碼添加到文檔導言區，位於 bib 模塊加載及其設置代碼區域的下面：

```
\unprotect
\setuppublicationlayout[booklet]{%
  \insertauthors{}{\unskip.}{}%
  \insertitle{ \bgroup\it }{\/\egroup\unskip.}{}%
  \insertbiurl{ URL: }{\unskip.}{}%
  \insertnote{ }{\unskip.}{}}
\protect
```

`\setuppublicationlayout` 命令是用於設置參考文獻列表的排版佈局的，上例中，其參數我設爲 booklet，這是因爲我在 .bib 文檔中將將網絡文檔類型定義爲 booklet（小册子）類型，它是 BibTEX 的標準文獻條目類型之一，將其作爲 `\setuppublicationlayout` 的參數，表示這裡要設置該類型文獻條目的排版風格。

在這份筆記中，我只需要 booklet 條目由 author（作者）、title（標題）、URL（文獻網址）以及 note（文獻註解）附加註釋信息構成，因此在設置 booklet 文獻條目的排版風格時，對於每一條目要素都對應一個 `\insertxxx` 命令。`\insertxxx` 命令有三個參數，可用於設定每一文獻條目構成要素的排版環境，前兩個參數分別用於設置對應排版文獻條目構成要素的前後排版命令，第三個參數用於設置對應的文獻條目構成要素預設狀態。以 `\insertauthors{}{\unskip.}{}` 爲例，第一個參數是空的，表示採用默認排版環境；第二個參數爲 `\unskip.`，表示消除 booklet 文獻條目中作者信息之後的空格，並且添加句點，第三個參數爲空，表示當 .bib 文件中的 booklet 條目未提供作者信息時不作任何處理。

注意，在上述示例中，\insertbiburl 命令是啓來 .bib 中所提供的文獻 URL 信息的。如果你希望文檔中的 URL 是超級鏈接格式，那麼還需要在 ConTEXt 文檔導言區添加以下代碼：

```
\setupinteraction[state=start] % 啓用文檔的可交互功能
```

利用上述方法並配合 ConTEXt 各種排版命令可以靈活地控制參考文獻列表的排版格式，這就是爲什麼 ConTEXt 的 bib 模塊只是將 BibTEX 的 .bst 文檔用來實現參考文獻列表排序的主要原因。

## 4.5 書籤

在閱讀很長的 PDF 文檔時，書籤 (Bookmark) 是很有用的，它們通常顯示於 PDF 閱讀器的側欄，用戶用鼠標點擊書籤就可以快速打開書籤所指向的頁面。對於如何使用 ConTEXt 實現書籤，由於我是沒有耐心的人，所以看到 ConTEXt Wiki 上提供了適合沒耐心的人對文檔書籤的生成快速建立初步認識的代碼，便直接抄過來：

```
% 啓用書籤功能
\setupinteraction[state=start]
\setupinteractionscreen[option=bookmark]
\placebookmarks[chapter][chapter]

\starttext
\chapter{飛刀與快劍}
冷風如刀，以大地爲砧板，視眾生爲魚肉。\par
萬里飛雪，將穹廬作洪爐，熔萬物爲白銀。\par
……
```

```
\chapter{海内存知己}
馬車裡堆著好幾罈酒，這酒是那少年買的，
所以他一碗又一碗地喝著，而且喝得很快。\par
……
\stoptext
```

用常規的 PDF 閱讀器打開所生成的文檔，若開啓側欄面板，應當可以看到書籤了。如果你看不到，那麼就見識一下圖 4.2。

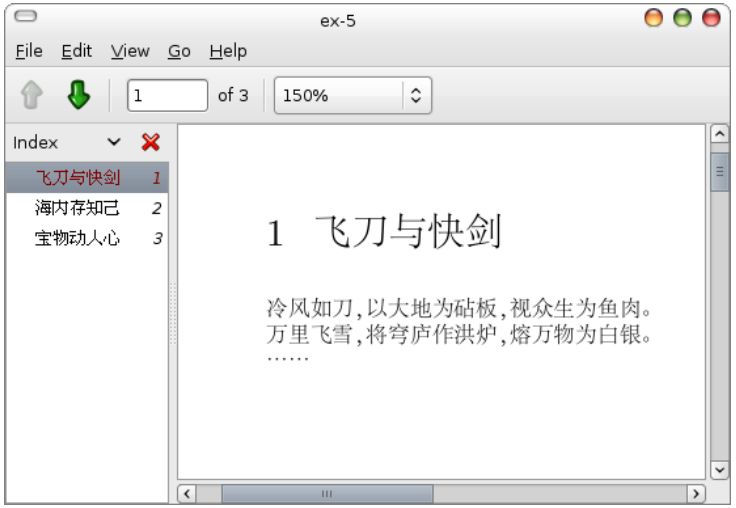



圖 4.2　帶書籤的 PDF 文檔

要開啓 ConTEXt 的書籤功能，必須 \setupinteraction[state=start]。另外，PDF 閱讀器的側欄的書籤窗口默認是隱藏的，要讓閱讀器打開文檔時自動開啓書籤窗口，那就是上例中 \setupinteractionscreen 命令所做的。

將章、節標題作爲書籤的想法很好，這樣在書籤窗口中，可以通過章節標題對文檔的大致內容有所瞭解。在上例中，我通過 \placebookmarks 命令設置 chapter 標題作爲書籤。\placebookmarks 的第一個參數是設定各層次的章節類型作爲書籤，第二個參數是設置默認可見的書籤。看下面一個更複雜的示例：

```
\placebookmarks[chapter,section,subsection][chapter]
```

這個示例的含義是：將章、節、小節的標題作爲書籤，默認只在書籤窗口中顯示由章標題生成的書籤。

在 MkIV 中，不知是特性還是缺陷，在將章節標題作爲書籤時，若標題中出現了格式化文本，譬如 \ConTeXt 這樣的文本，那麼書籤中就會將 \ConTeXt 的 TEX 定義顯示出來，而不是顯示 ConTeXt 或者 ConTEXt。現在，我未能找到解決這一問題的好方法，笨方法是有的。在撰寫某章節標題時，若是需要使用格式化文本，那麼就在章節命令之後使用 \bookmark 命令手動設定書籤標題：

```
\section{安裝 \ConTeXt\ Minimals}
\bookmark{安裝 ConTeXt Minimals}
```

本文檔目前正是使用這種方法去除書籤中出現的格式化文本，勉強對付過去。

# 參考文獻


[1] PRAGMA. *ConTEXt an excursion*. URL: http://www.pragma-ade.com/general/manuals/ms-cb-en.pdf.

[2] PRAGMA. *ConTEXt the manual*. URL: http://www.pragma-ade.com/general/manuals/cont-eni.pdf.

[3] PRAGMA. *Fonts in ConTEXt*. URL: http://context.aanhet.net/svn/contextman/context-reference/en/co-fonts% .pdf.

[4] Hans Hagen. *METAFUN*. URL: http://www.pragma-ade.com/general/manuals/metafun-s.pdf.

[5] Nicolas Markey. *Tame the BeaST: The B to X of BibTEX*. URL: http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/.

[6] Taco Hoekwater. *Bibliographies*. URL: http://modules.contextgarden.net/bib.

[7] Taco Hoekwater. *Bibliography*. URL: http://wiki.contextgarden.net/Bibliography.

色即是空